REMARKS

Claim 1 has been amended. Claims 1-43 and 46-59 remain pending in the application. Reconsideration is respectfully requested in light of the following remarks.

Section 102(a) Rejection:

The Examiner rejected claims 1-15, 17-32, 34-43, 46-49 and 51-59 under 35 U.S.C. § 102(a) as being anticipated by Daynes (U.S. Patent 6,182,186). Applicants traverse this rejection for at least the following reasons.

Regarding claim 1, contrary to the Examiner's assertion, Daynes does not disclose *a method of providing non-blocking multi-target transactions in a computer system*. The Examiner previously cited FIG. 7 and related text in his remarks. However, nothing in Daynes describes non-blocking multi-target transactions. Instead, Daynes describes the use of a respective lock state to control access to each of a plurality of resources. FIG. 7 of Daynes illustrates the process of acquiring a lock for a single resource. This process includes placing a lock request in a queue until the lock becomes available (e.g., until it is released by another transaction), which is clearly a blocking mechanism, as would be understood by those of ordinary skill in the art. Daynes describes a non-blocking synchronization for changing the lock state associated with a single one of these resources (see, e.g., column 17, lines 2 – column 18, line 13). However, this clearly does not provide a non-blocking multi-target transaction, since this mechanism is used to access a single target, the lock state associated with a single shared resource.

In the Final Action mailed July 18, 2008, the Examiner cites additional passages in Daynes as teaching this limitation (e.g., FIG. 13; column 15, lines 64-67; and column 16, lines 1-10). This passage describes that two different techniques may be employed in the implementation of a locking operation, which is clearly a blocking operation. This passage describes that one of the techniques used by a locking operation is a non-blocking synchronization. The other is dispatching of specialized code. FIG. 13

illustrates code for dispatching the execution of the locking operation dependent on the lock state's type. This code includes a non-blocking synchronization primitive (in this example, a CAS instruction). As discussed above, this synchronization primitive is used within the locking operation to change the value of the lock state. **The use of a non-blocking synchronization primitive *within* a locking operation does not change the fact that the overall locking operation is a *blocking* operation on a *single* resource managed by the lock. This clearly cannot and does not teach *a method of providing non-blocking multi-target transactions*, as recited in Applicants' claim.**

In the Response to Arguments section of the Final Action, the Examiner again cites column 17 (lines 25-40) as teaching the non-blocking multi-target transactions of Applicants' claim. Applicants again assert that this passage teaches the use of a non-blocking synchronization primitive within a **locking** operation, as discussed above. The Examiner also cites column 19, lines 15-25. This passage describes that bit numbers used to identify locking contexts may be recycled through the use of garbage collection. If there are no bits mapped to locking contexts of terminated transactions that did not delete their bits from the lock states represented by the locks these transactions owned upon their completion, garbage collection proceeds as usual. Applicants assert that this describes yet another technique that may be employed in a system that employs a blocking mechanism (i.e., locks) to control access to resources. The Examiner submits, "Therefore Daynes teaches non-blocking mechanism used to access multiple transactions." Applicants note that claim 1 does not recite a "non-blocking mechanism used to access multiple transactions," nor is it clear what the Examiner means by this phrase. In addition, there is nothing in Daynes that teaches that non-blocking synchronization primitives are used to change the state of multiple locks, as the Examiner seems to be implying. Instead, in Daynes, each resource is managed by a respective lock, and each instance of such a primitive may be used to change the lock state of one lock. Finally, even if a non-blocking synchronization primitive were used to change the state of multiple locks, this would not teach the non-blocking multi-target transaction of Applicants' claim, as the transaction itself (which uses locks to manage access to resources) is clearly a blocking operation.

Further regarding claim 1, Daynes does not disclose *defining plural transactionable locations, wherein individual ones of the transactionable locations encode respective values and are owned by no more than one transaction at any given point in a multithreaded computation*. The Examiner cites FIGs. 4 and 11 (specifically, element 1148) and column 18, lines 15-25 as teaching these limitations. The passage cited by the Examiner describes that one type of lock that a transaction may own is an exclusive lock (i.e., a single-write-owner, or SWO, lock). However, Daynes clearly describes other lock states (e.g., MRO: multiple read owner, and MWO: multiple write owner) that represent ownership by multiple owners for a given resource. FIG. 4, also cited by the Examiner, actually illustrates this multi-owner concept and, thus, **teaches away from the above-referenced limitation of claim 1**. In FIG. 4, element 408 represents a lock state with an empty write set and a read owner set made up of two transactions, T_1 and T_2. Element 1148 of FIG. 11, also cited by the Examiner, also supports this multi-owner functionality. This element represents a global variable comprising a set of transactions that may be ignored by other transactions in cases that would otherwise result in an ownership conflict (e.g., allowing multiple owners to be included in a read or write set, but causing a terminated transaction that has not yet been removed from the read or write set to be ignored).

In the Response to Arguments section of the Final Action, the Examiner cites column 2 of Daynes, which describes one type of lock referred to as an exclusive lock. Applicants note that this passage is part of the Background section of the invention, and does not describe a feature of any embodiment of Daynes relied on in teaching other limitations of Applicants' claims. Instead, this passage provides a general description of shared locks versus exclusive locks. **Applicants again assert that any use of locks provides a blocking mechanism and, therefore, teaches away from Applicants' claimed invention.**

Daynes also fails to disclose *wherein the ownership acquiring wrests ownership from another transaction, if any, that owns the targeted transactionable location without*

the other transaction releasing ownership. The Examiner previously cites column 18, lines 45-50 as teaching *wresting ownership from another transaction*. This passage describes the memory usage of the lock states of Daynes invention and has nothing to do with the above-referenced limitation of claim 1. Applicants assert that Daynes clearly does not teach that one transaction wrests ownership from another transaction that owns a targeted transactionable location, as recited in the claim. Instead, as illustrated in FIG. 7 and FIG. 13, a lock may not be acquired by one transaction if there is a conflict with another transaction (e.g., if another transaction holds an exclusive lock such as an SRO or SWO lock). In this case, a new lock request is placed in a queue until the conflict is resolved (e.g., the owning transaction releases the lock) and until any lock requests already pending in the queue are serviced ahead of the new lock request. In the case that a multi-owner lock state is associated with a resource, an additional lock may be acquired. However, in this case, ownership is not wrested from any other owners. **Since in neither of these cases, nor any other scenario described in Daynes, a transaction wrests ownership from another transaction, Daynes clearly does not disclose the above-referenced limitation.**

In the Response to Arguments section of the Final Action, the Examiner cites FIG. 9 and column 13, lines 40-60, which includes, "At step 904, for each lock state found, the transaction is removed from all owner sets where it appears. By removing the transaction from the owner set, the value of the lock state is modified." The Examiner submits, "By removing the transaction from the owner, the ownership acquiring wrests ownership from another transaction." Applicants assert that this passage describes the removal of a transaction from an owner set in response to the transaction ending (shown as 900 in FIG. 9), not in response to ownership being wrested away from one transaction for another (ownership acquiring) transaction, as in Applicants' claim. In various embodiments of Daynes, a transaction that has ended releases its locks, removes itself from an owner set, or is removed from an owner set by a lock manager in response to the transaction ending. **Nothing in Daynes teaches ownership acquiring, whereby ownership is wrested away from another transaction that owns a transactionable location, as required by Applicants' claim.**

Finally, Daynes fails to disclose *attempting to commit the particular multi-target transaction using a single-target synchronization primitive to ensure that, at the commit, the particular multi-target transaction continues to own each of the targeted transactionable locations*. The Examiner cites column 19, lines 1-15 and column 20, lines 5-15 in teaching these limitations. The Examiner's citation in column 19 describes that inactive bit numbers may be recycled (e.g., used in a subsequent owner set). **Applicants again assert that this has absolutely nothing to do with the above-referenced limitation of claim 1.** The Examiner's citation in column 20 describes how lock states may be cached and the cached lock states may be used for acquiring the lock of an unlocked resource. **It also has nothing to do with the above-referenced limitation of claim 1.** In addition, as discussed above, the only single-target synchronization primitive described in Daynes is used to attempt to update the lock state associated with a given resource, not to commit a multi-target transaction. Applicants assert that nothing in Daynes discloses the use of a single-target synchronization primitive in an attempt to commit a multi-target transaction, as required by claim 1.

In the Response to Arguments section of the Final Action, the Examiner cites FIG. 7 and its description as teaching the above-referenced limitation. **This figure illustrates a process for lock acquisition using a single-target synchronization primitive (compare-and-swap). It has absolutely nothing to do with an attempted commitment of a non-blocking multi-target transaction using such a primitive.** The additional citation of FIG. 7 and its description clearly do not teach this limitation.

For at least the reasons above, Daynes cannot be said to anticipate claim 1 and removal of the rejection thereof is respectfully requested.

Independent claims 22, 46, and 56 include limitations similar to those discussed above regarding claim 1. Therefore, the arguments presented above apply with equal force to these claims, as well.

Applicants note that the Examiner has again failed to include any remarks directed to Independent claim 46 in the Office Action, although claim 46 is listed in the rejection under 35 U.S.C. § 102(a). Therefore, no *prima facie* rejection has been stated for claim 46.

Section 103(a) Rejection:

The Examiner rejected claims 16, 33 and 50 under 35 U.S.C. § 103(a) as being unpatentable over Daynes in view of Maged M. Michael, et al. ("Non-Blocking Algorithms and Preemption-Safe Locking on Multiprogrammed Shared Memory Multiprocessors". Applicants traverse the rejection of claims 16, 33 and 50 for at least the reasons given above in regard to the claims from which they depend.

In regard to the rejections under both § 102(a) and § 103(a), Applicants assert that numerous ones of the dependent claims recite further distinctions over the cited art. However, since the rejections have been shown to be unsupported for the independent claims, a further discussion of the dependent claims is not necessary at this time. Applicants reserve the right to present additional arguments.

CONCLUSION

Applicants submit the application is in condition for allowance, and an early notice to that effect is requested.

If any fees are due, the Commissioner is authorized to charge said fees to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/6000-33600/RCK.

Respectfully submitted,

/Robert C. Kowert/
Robert C. Kowert, Reg. #39,255
Attorney for Applicants

Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8850

Date: October 20, 2008